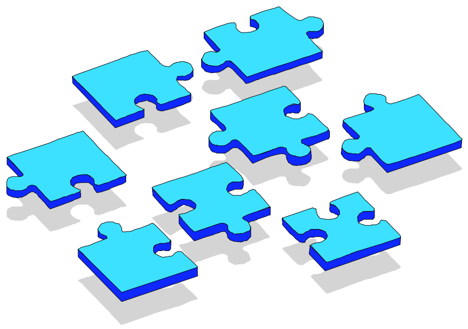
BlueGene/L Software Overview

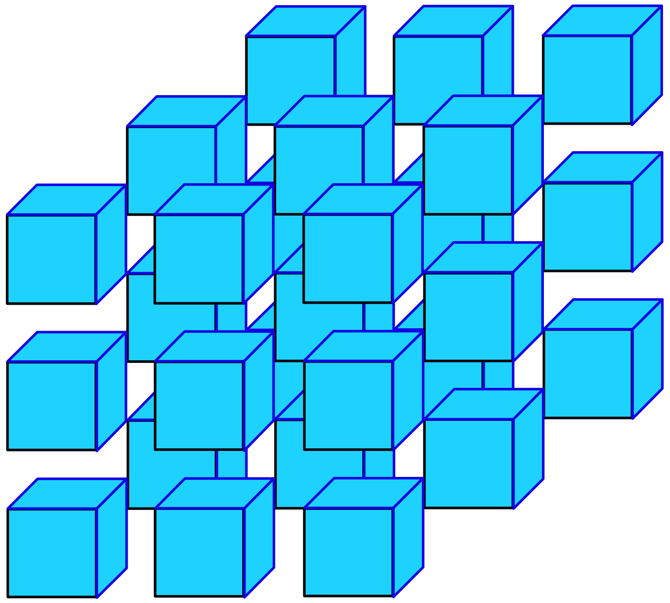
Sid Chatterjee Manish Gupta Jose Moreira
IBM T. J. Watson Research Center

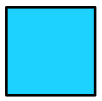


Outline

- System software overview
  - Operating system - Linux, HPK
  - Compilers - Fortran95, C99, C++
  - Math library - subset of ESSL
  - Message passing - MPI
  - File system
  - Job scheduler
  - System management, including control, bringup, and RAS
- In-depth look at some software components
  - Single node compilation and performance issues - Sid Chatterjee
  - Message passing support: design and performance issues - Bill Gropp, Rusty Lusk, Jose Moreira

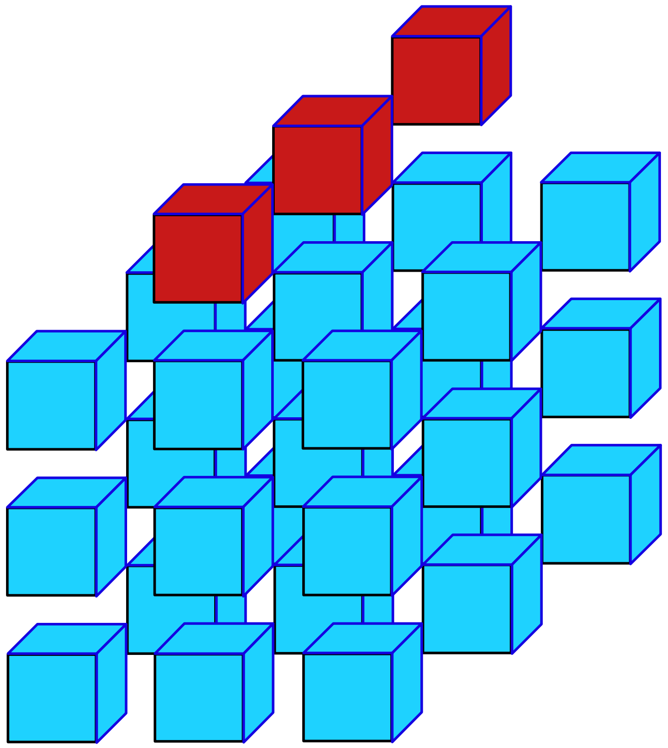
BG/L: Application Developer's View





- Collection of compute nodes with fast network connections
- Compute nodes dedicated to running user application, and almost nothing else - simple high performance kernel (HPK)

BG/L: Operating System Services

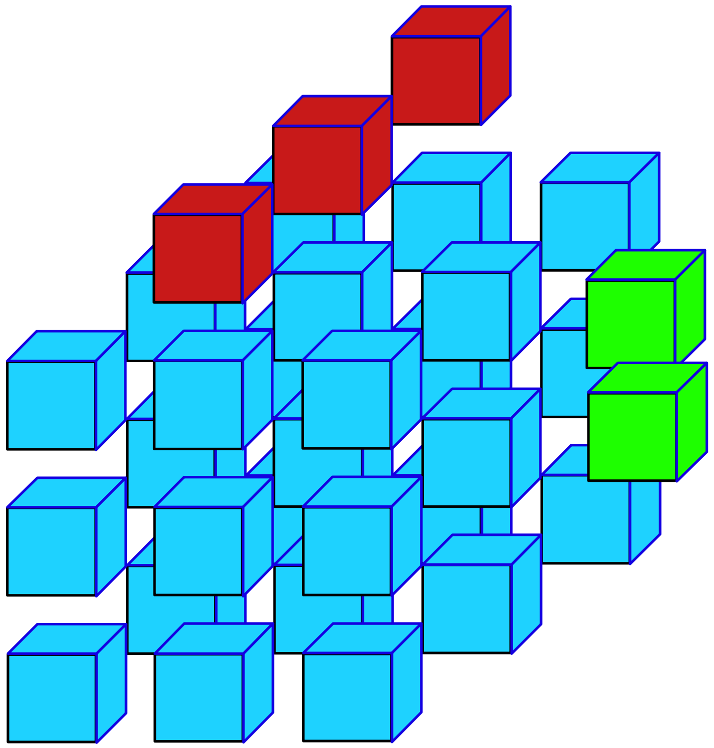


🟦 compute node
application volume

🟥 I/O node
operational surface

- I/O nodes provide a more complete range of OS services, e.g., I/O, sockets, process launch and termination - run Linux
- Operationally, compute nodes are "invisible" - allows OS on compute nodes to be simple, makes the machine more manageable

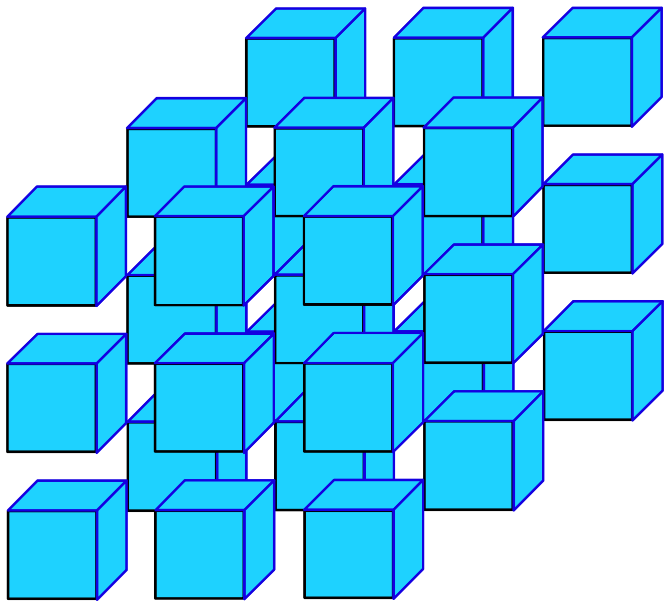
BG/L: Overall Operational View



🟦	compute node application volume
🟥	I/O node operational surface
🟩	service node control surface

- Service nodes have private ethernet connection to core BG/L components
- System management services (e.g. heartbeating, monitoring errors) largely transparent to application/system software

BG / L: Programming Models



🟦 N compute nodes

Default mode of operation

- SPMD with message passing (MPI) among N compute processes
  - one processor on each node dedicated to communication
- Foundation for other programming models, e.g., UPC, Charm++

Other Modes: Using Both CPUs for Computation

- Message passing between the two CPUs on a compute node
  - use two MPI processes on each node, one bound to each processor
  - each processor on a node handles both computation and communication
  - distributed memory semantics convenient on node w/o cache coherence
- Shared memory programming on compute node
  - Can be selectively used for compute-bound parallel regions
  - Need careful sequence of cache line flush, invalidate, and copy operations to deal with lack of L1 cache coherence in hardware
  - Alternatives:
    - OpenMP pragma extension - compiler managed code generation
    - Explicit multithreading - user has to manage coherence of caches
- Caveat - these modes may not lead to higher performance if program is communication bound or memory bandwidth bound

High Performance Kernel

- Simple kernel for compute nodes
- Support for:
  - multithreaded (fixed number of threads), single process execution
  - simple memory management for a fixed size virtual memory space
    - no demand paging
    - no TLB misses - using large pages
    - set attributes on pages to control cache behavior
  - interaction with control system
  - I/O through function shipping to I/O nodes
  - debugging through ptrace client
- Same set of user-space services as Linux, including: libraries, compilers, system programming interface (SPI)

Linux

- BG/L specific extensions to PPC 440 port
  - device drivers for ethernet, tree
  - interrupt controller
  - support for double FPU
- Enhancements
  - smart device drivers for exploiting second CPU
  - SMP mode without cache coherence
  - support for variable-sized pages
  - demand paging over network
  - eliminate unnecessary daemons
  - reduce frequency of asynchronous events

BGLsim - System-level simulator (1)

- Program development based on complete system-level simulator for BG/L
  - based on SimOS-like Mambo simulation framework from Austin Research Lab.
  - complements our VHDL simulators (which are much slower)
- This simulator is architecturally accurate:
  - executes the full instruction set of BG/L, including SIMD Floating Point Unit
  - models all the devices, including Ethernet, torus, tree, lock box, etc
  - supports development of system software and applications
  - currently not performance aware: only counts instructions, not cycles
  - we have simulated systems with up to 128 nodes (not a limitation)
- Efficient simulation that supports code development:
  - 500,000 - 1,000,000 BG/L instructions/second on 1 GHz Pentium III
  - generates statistics like instruction counts, cache misses.
- We have used it to develop/test HPK, Linux, device drivers, networking, MPI, compilers, FPU benchmarks

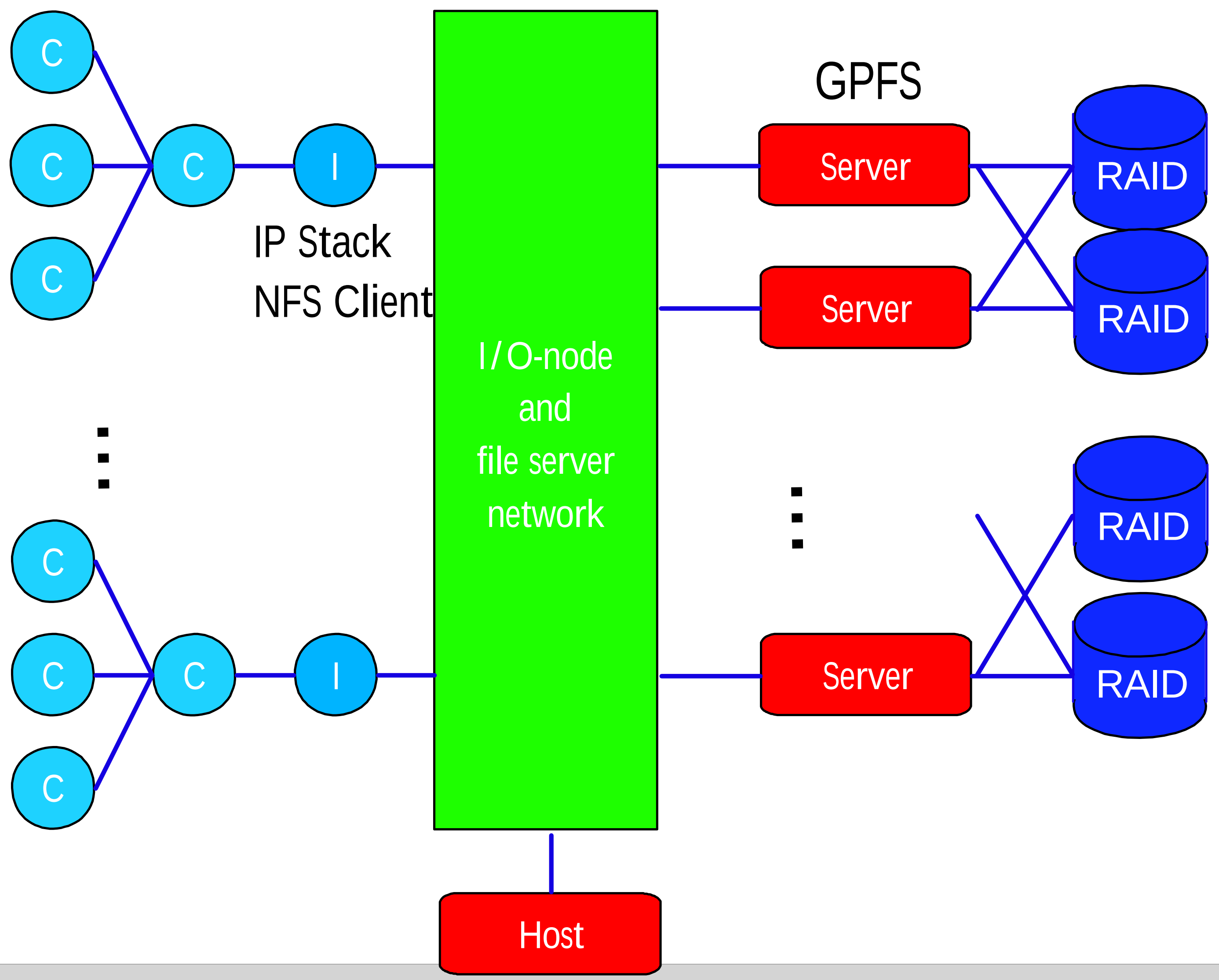
Node Compilers for BlueGene/L

- Support Fortran, C, C++
- Backend enhanced to support PPC440 and to target SIMD FPU on nodes
  - Finds parallel operations that match SIMD instructions
  - Register allocator enhanced to handle register pairs
  - Instruction scheduling tuned to unit latencies
- Initial design of (2-way) SIMD FPU architecture was driven by key workload kernels such as matrix-matrix product and FFT
  - Identified several mux combinations for SIMD operations not usually seen on other SIMD ISA extensions (Intel SSE, PPC AltiVec), e.g.,
    $d_P = a_P + b_P * c_P \;||\; d_S = a_S - b_S * c_S$
    $d_P = a_P + b_P * c_P \;||\; d_S = a_S + b_P * c_S$

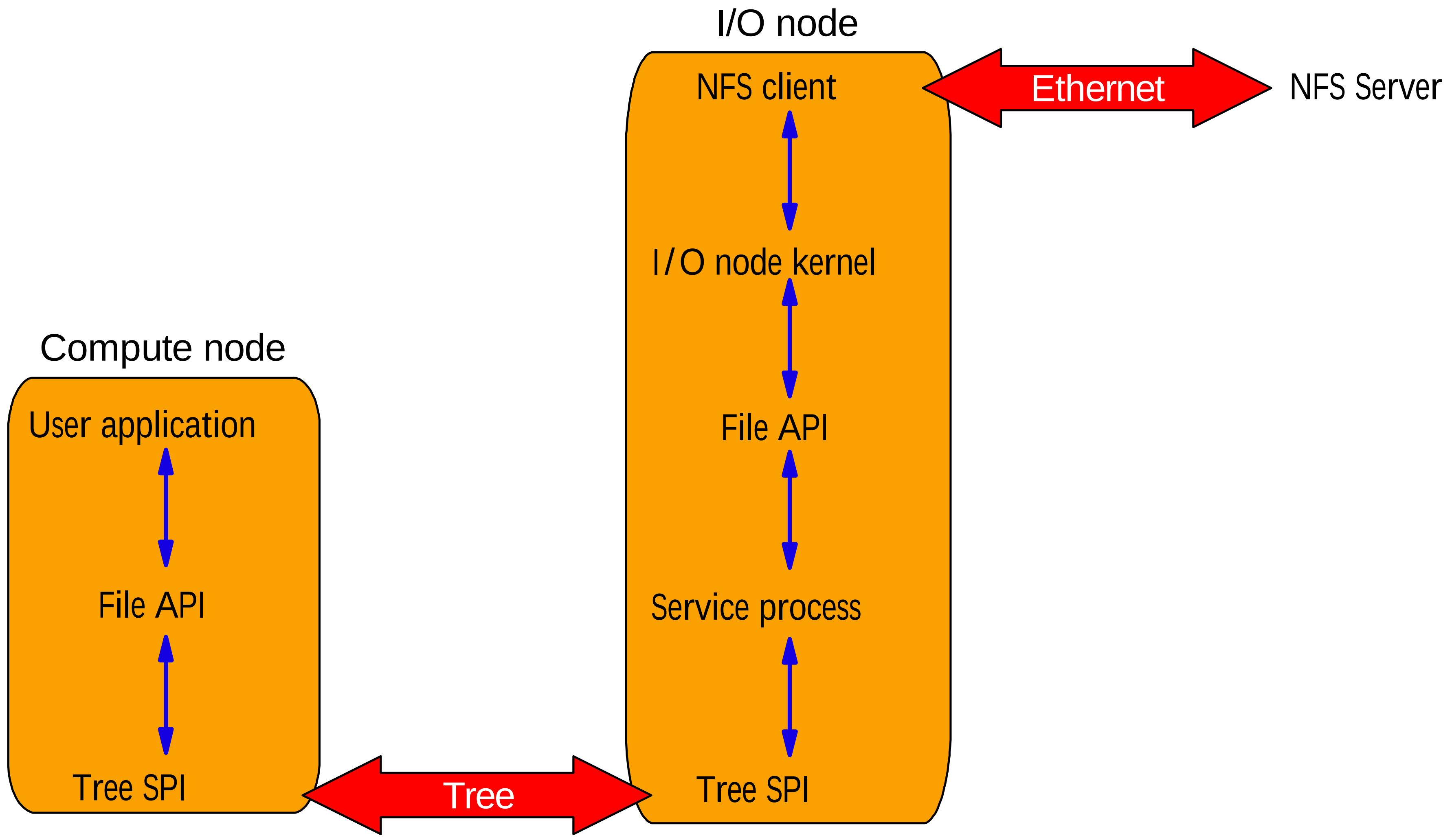
Message Passing

- Three layers of communication libraries:
  - HAL - delivery of packets
  - Message layer - delivery of arbitrary sized messages
  - MPI - for end user application
- HAL and Message layer contain BG/L-specific code
  - delivery of packets over torus and tree networks
  - using second CPU on compute node as communication coprocessor
  - can be used by applications, but are intended more for library developers
- MPI based on MPICH2 being developed by Argonne National Lab
  - builds upon Message layer
  - need highly scalable implementation for tens of thousands of nodes - consistent with MPICH2 design goals
  - BG/L specific optimizations for exploiting tree and broadcasts on torus

File System: GPFS on PC Cluster



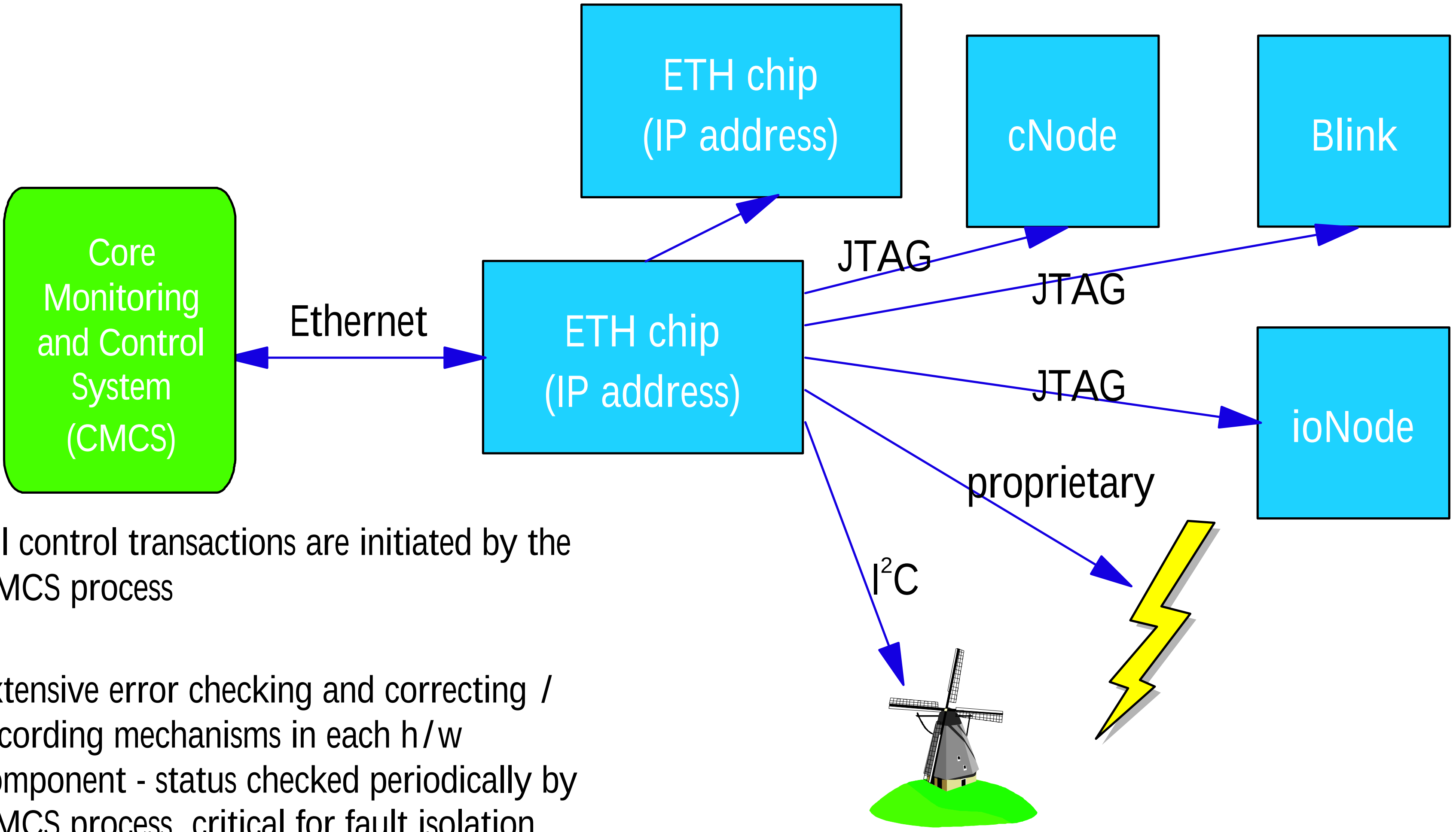
NFS file I/O in BG/L



Tools

- Debugger
  - Preliminary discussions with Etnus to support TotalView debugger on BG/L
  - We will provide *ptrace* debug client support on BG/L nodes.
- Performance tools
  - Possibilities for visualization of messaging traffic are: Paravar (UPC Barcelona), Vampir (PALLAS), PE Benchmarker (IBM)
  - Performance counters data : PAPI

Control network



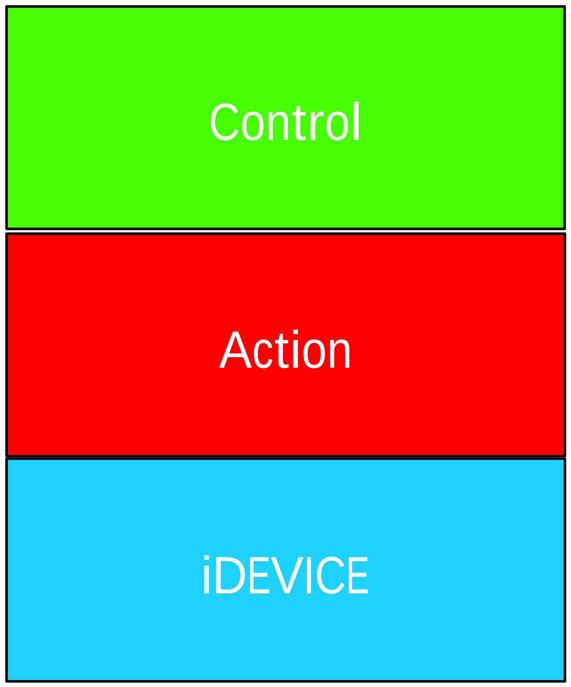
All control transactions are initiated by the CMCS process

Extensive error checking and correcting / recording mechanisms in each h/w component - status checked periodically by CMCS process, critical for fault isolation.

Each device under control is identified by (IP, device number)

Core Monitoring and Control System (CMCS)

- iDEVICE layer:
  - a transport layer for talking to devices over IP
  - sends / receives bit-vectors
- Action layer:
  - implements device-specific semantic actions
  - straightforward correspondence to bit-vectors at iDEVICE layer
- Control layer:
  - implements complex operations with multiple steps
  - E.g., IPL, power on, etc



Overall System Management

- CMCS fits into the Cluster System Management (CSM) framework as a Resource Manager
- CSM framework provides
  - high availability services
  - connectivity between components
  - monitoring services
  - GUI for system administrator
- Extending with RAS database containing information about:
  - machine topology (compute nodes and I/O nodes)
  - IP address of each ETH and devices attached to it
  - state (assumed and/or measured) of each device
  - bitvectors for controlling devices
  - event logs - information on errors

Fault Recovery

- Application-level checkpointing
  - user controlled - application does self checkpointing
  - application responsible for quiescing before checkpoint
  - implementation in user level library
  - restart is user-initiated
- System-initiated checkpointing
  - will explore different levels of transparency

Job Scheduling

- Job scheduling strategies can significantly impact the utilization of large computer systems
- Machines with toroidal topology, as opposed to all-to-all switch, are particularly sensitive to job scheduling
- Based on IBM LoadLeveler product
  - User submits jobs from the host (Front End node)
  - Support for initiating parallel jobs on BG/L core based on MPICH2
- BG/L specific extensions
  - Topology aware scheduling, including backfilling
  - Task migration

Conclusions

- We are developing a BG/L system software stack with Linux-like personality for user applications
  - custom solution (HPK) on compute nodes for highest performance
  - Linux solution on I/O nodes for flexibility and functionality
- We will exploit both CPUs in the BG/L compute node
- Communication infrastructure is layered - supports MPI and the development of new application-level communication libraries
- Simulator supports system software and application development
- Leveraging IBM product infrastructure on compilers, job scheduling/management (LoadLeveler) and clustered systems management (CSM)